PROCEEDINGS OF THE SIXTH SIAM CONFERENCE ON

PARALLEL PROCESSING FOR SCIENTIFIC COMPUTING

Volume II

Edited by Richard F. Sincovec
Oak Ridge National Laboratory
David E. Keyes
Yale University
Michael R. Leuze
Oak Ridge National Laboratory
Linda R. Petzold
University of Minnesota
Daniel A. Reed
University of Illinois

siam.

Philadelphia

Society for Industrial and Applied Mathematics

Parallel Preconditioning and Approximate Inverses on the Connection Machine*

Marcus Grote[†] Horst D. Simon[‡]

Abstract

We present a new parallel approach to preconditioning for very large, sparse, unsymmetric, linear systems. We explicitly compute an approximate inverse to our original matrix. This new preconditioning matrix can be applied most efficiently for iterative methods on massively parallel machines, since the preconditioning phase involves only a matrix-vector multiplication, with possibly a dense matrix. Furthermore the actual computation of the preconditioning matrix has natural parallelism. For a problem of size n, the preconditioning matrix can be computed by solving n independent small least squares problems. The algorithm and its implementation on the Connection Machine CM-2 are discussed in detail and supported by extensive timings obtained from real problem data.

1 Introduction

Up to today, preconditioning methods on massively parallel systems have faced a major difficulty. The most successful preconditioning methods in terms of accelerating the convergence of the iterative solver such as incomplete LU factorizations are notoriously difficult to implement on parallel machines for two reasons: (1) the actual computation of the preconditioner is not very floating-point intensive, but requires a large amount of unstructured communication, and (2) the application of the preconditioning matrix in the iteration phase (i.e. triangular solves) are difficult to parallelize because of the recursive nature of the computation.

We shall consider the numerical solution of very large but sparse linear systems of the form

$$Ax = b, \quad x, b \in R^n$$

without assuming any special properties for A such as symmetry or definiteness. The order of A typically lies between, say, one thousand and one million. However, A usually has just a few nonzero elements per column. Our aim is to construct an approximate inverse M to A and consider applying the iterative solver to the *preconditioned* system

$$AMy = b, \quad M \approx A^{-1}, \quad x = My \; .$$

*The work of the second author was supported through NASA Contract NAS 2-12961.
[†]Stanford University, Stanford, CA 94305, (grote@sccm.stanford.edu)
[‡]Computer Sciences Corporation, Mail Stop T045-1, NASA Ames Research Center, Moffett Field, CA 94035, (simon@nas.nasa.gov)

2 Overview of the algorithm

The CM-2 is a massively parallel SIMD machine, and can, as our preliminary timings show, compute Mv extremely rapidly when M is a banded matrix. Thus it seems natural to require that M be a matrix with , say, $2p+1$ diagonals, $p \geq 0$, so that we may keep t_m very small relatively to the time spent for computing Av.

The closeness might be measured in some norm $\| \|$, so that we need to find an M which minimizes

$$\|AM - I\|.$$

In general, this problem is even harder than solving $Ax = b$; the main idea of our approach is to choose the norm to be the Frobenius norm.

Let us denote by A_k the columns of A. It is relatively simple to realize that to minimize

$$\|AM - I\|_F,$$

we need to minimize

$$\|AM_k - e_k\|_2$$

for each k individually, $1 \leq k \leq n$. Since each M_k contains at most only $2p+1$ nonzero elements, we need to solve n independent least square problems of size only $n \times (2p+1)$. This can be done by constructing, factorizing and solving all the normal equations simultaneously with parallelism of order n, where n is the dimension of the original system. To construct the normal equations, we must compute the sparse inner products of each column A_k with itself and its $2p$ *nearest* neighbors. For more details see [2].

Faced with a plethora of iterative methods for solving nonsymmetric linear systems, we opted for Van der Vorst's algorithm Bi-CGSTAB [6]. The method is transpose free and easy to implement. We apply Bi-CGSTAB to the preconditioned system

$$AMy = b$$

and simply multiply each vector by M before it is multiplied by A. The solution x_n is then merely given by $x_n = My_n$. These additional multiplications by M are extremely fast on the CM-2 because of the banded structure of M. We choose $x_0 = y_0 = 0$ as an initial guess. Our stopping criterion for a given tolerance $\epsilon > 0$ is $\frac{|r_n|}{|r_0|} < \epsilon$, where $r_0 = b - AMy_0 = b$ and $r_n = b - AMy_n$.

3 Results

3.1 Convergence

The test matrices we used are the SHERMANx and PORESx matrices from the Boeing-Harwell sparse matrix collection. To evaluate the effectiveness of the preconditioner in terms of convergence, we compare our results to the results reported by Tong in [5]. We use the same stopping criterion with $\epsilon = 10^{-8}$ and iterate up to a maximum of 2000 iterations. Tong uses incomplete ILU(k) and ILUT(k) preconditioning where k is small and chosen in an optimal way. This is a standard preconditioner but inherently very difficult to parallelize and very expensive for general sparse matrices. We see that although we are able to reduce the number of iterations by a factor up to 10, we usually do not perform as well as the ILU preconditioner in terms of number of iterations.

	P1	P3	S1	S3	S4	S5
Unpreconditioned[5]	200	1698	384	*	110	1846
Preconditioned[5]	15	17	18	66	25	15
Unpreconditioned	248	*	403	*	92	*
Precond. 1 diag.	B	441	224	406	69	883
Precond. 3 diag.	378	521	81	339	56	711
Precond. 5 diag.	243	926	68	342	56	240
Precond. 11 diag.	170	*	65	375	55	235
Precond. 21 diag.	36	*	46	340	55	222
Precond. 35 diag.	7	*	45	340	47	212
Precond. 51 diag.	2	B	42	335	46	215

* : more than 2000 iterations
B : Breakdown in Bi-CGSTAB

TABLE 1

Study of convergence for poresx and shermanx

	Unpreconditioned		Preconditioned		
	Nb. of iter.	Exec. time[s]	Nb. of diag. in M	Nb. of iter.	Minim. exec. time[s]
S1	127	2.5	5	21	0.62
S3	*	*	5	206	12.4
S4	62	1.6	3	36	1.1
S5	*	*	9	120	9.3
P1	286	5.1	23	9	0.92
P3	*	*	7	361	8.77

* : more than 500 iterations
B : Breakdown in Bi-CGSTAB

TABLE 2

Study of overall performance

3.2 Number of diagonals vs. total execution time

Here the tolerance $\epsilon = 0.001$ and the maximum number of iterations is 500. All the runs were done on a 8K processor CM-2, in double precision and under slicewise.

In table 2 we computed the number of diagonals in M which minimizes the total execution time by increasing the number of diagonals up to 51.

4 Performance analysis on a large problem

We used a large problem with about 16,000 unknowns to study the performance of the implementation on the CM-2. The matrix is a model of the HSCT (High Speed Civil Transport plane) obtained from Olaf Storaasli at NASA Langley Research Center. We are intersted in the performance of the construction of M, the multiplication by M and A, and in the overall performance of the algorithm.

The multiplication with A is done with the sparse-matvec-mult routine provided by

Nb. of diagonals in M	1	3	5	7	9	11
Compute M time[%]	0.3	0.6	0.9	1.2	1.5	1.8
Compute M [Mflops/sec.]	10.8	15.7	17.9	19.7	21.4	23.4
$A \times v$ time[%]	98.8	98.3	97.7	97.2	96.6	96.1
$A \times v$ [Mflops/sec.]	16	16	16	16	16	16
$M \times v$ time[%]	0.07	0.3	0.6	0.8	1.0	1.3
$M \times v$ [Mflops/sec.]	341	239	225	221	218	214
Overall effective Mflops/sec.	17.4	17.8	18.3	18.7	19.2	19.6

TABLE 3

Performance analysis for olaf

the CMSSL scientific software library — we note here that the *trace* (a variable which stores the sparsity structure of A) is computed only once and then saved for additional multiplications. The multiplication with M is easy since M is banded and requires local (CSHIFT) communications only. All the dimensions were always extended up to a multiple of 4096 in order to enhance the performance — the CSHIFT command is very sensitive in that respect. Yet, the overall *effective* Mflops rates were computed using the number of *necessary* operations only. All these executions ran for 500 iterations.

In table 3 which was obtained on a 32K CM-2, the time percentages are always given with respect to the total execution time. We observe that the multiplications by M are negligible and very efficient. The computation of M necessitates indirect addressing which is rather costly, but since M is computed only once, the time spent to compute it remains neglegible. The overall relatively low performance hinges upon the 16 Mflop rate at which the sparse matrix vector multiply routine of the CMSSL library operates.

5 Conclusions

Preconditioning the linear system by an approximate banded inverse M not only enhances convergence, but also reduces on the CM-2 the total execution time which includes the time necessary to compute M. It is a somewhat blunt tool in the sense that blindly increasing the number of diagonals in M does not significantly ameliorate the convergence beyond a certain point, typically somewhere between five and eleven diagonals. The computation of M does not necessitate any global communications on a SIMD machine such as the CM-2, and applying M to a vector v is extremely fast on a massively parallel machine.

References

[1] O. Axelsson. *A Survey of Preconditioned Iterative Methods for Linear Systems of Algebraic Equations*, BIT 25, 166-187, 1985.

[2] M. Grote and H. Simon, *Parallel Preconditioning and Approximate Inverses on the Connection Machine*, Proceedings of the Scalable High Performance Computing Conference (SHPCC) 1992, Williamsburg, VA, April 1992, IEEE Computer Science Press, 1992, pp. 76 - 83.

[3] S. L. Johnsson & K. K. Mathur. *Experience with the Conju gate Gradient Method for Stress Analysis on a Data Parallel Supercomputer.*

[4] L. Yu. Kolotilina & A. Yu. Yeremin. *Factorized Sparse Approximate Inverse Preconditionings*

[5] Charles H. Tong. *A Comparative Study of Preconditioned Lanczos Methods for Nonsymmetric Linear Systems.* Sandia report SAND91-8240, Sandia National Laboratories, January 1992.

[6] Henk A. Van der Vorst. *Bi-CGSTAB: A Fast and Smoothly converging variant of BI-CG for the Solution of Nonsymmetric Linear Systems.* Preprint, University of Utrecht, The Netherlands, September 1990.